TP 5 - éléments de correction

```python
import numpy as np
import matplotlib.pyplot as plt

# Fonctions pour les tests
def P_0(x):
    return np.ones_like(x)
def P_1(x):
    return x
def P_2(x):
    return x**2
def P_3(x):
    return x**3
def P_4(x):
    return x**4
def P_5(x):
    return x**5
def inv(x):
    return 1/x

# Routine rectangles à gauche
def rect1(f,a,b,n):
    x = np.linspace(a,b,n+1)
    s = 0
    for k in range(0,n):
        s += f(x[k])
    return (b-a)/n * s

def rect2(f,a,b,n):
    x = np.linspace(a,b,n+1)
    s = np.sum(f(x[0:n]))
    return (b-a)/n * s

# Routine point du milieu
def ptmilieu1(f,a,b,n):
    x = np.linspace(a,b,n+1)
    z = np.zeros(n)
    for k in range(n):
        z[k] = 0.5 * (x[k] + x[k+1])
    s = 0
    for k in range(n):
        s += f(z[k])
    return (b-a)/n * s

def ptmilieu2(f,a,b,n):
    x = np.linspace(a,b,n+1)
    z = (x[0:n] + x[1:n+1])/2
    s = np.sum(f(z))
    return (b-a)/n * s

# Routine trapèze
def trap1(f,a,b,n):
    x = np.linspace(a,b,n+1)
    s = 0
    for k in range(1,n):
        s += f(x[k])
    return (b-a)/(2*n) * (f(a) + 2*s + f(b))

def trap2(f,a,b,n):
    x = np.linspace(a,b,n+1)
    s = np.sum(f(x[1:n]))
    return (b-a)/(2*n) * (f(a) + 2*s + f(b))

# Routine Simpson
def simpson1(f,a,b,n):
    h = (b-a)/n
    x = np.linspace(a,b,n+1)
    s1 = np.sum(f(x[1:n]))
    xkp12 = (x[0:n] + x[1:n+1])/2
    s2 = np.sum(f(xkp12))
    return h/6 * (f(a) + f(b) + 2*s1 + 4*s2)
```

```python

def simpson2(f,a,b,n):
    h = (b-a)/n
    x = np.linspace(a,b,n+1)
    s1 = np.sum(f(x[1:n]))
    xkp12 = (x[0:n] + x[1:n+1])/2
    s2 = np.sum(f(xkp12))
    return h/6 * (f(a) + f(b) + 2*s1 + 4*s2)

# Erreurs pour tracé
def erreur_rect2(f,vec_n):
    return np.array([abs(rect2(f,1,2,2*n) - np.log(2)) for n in vec_n])

def erreur_ptmilieu2(f,vec_n):
    return np.array([abs(ptmilieu2(f,1,2,2*n) - np.log(2)) for n in vec_n])

def erreur_trap2(f,vec_n):
    return np.array([abs(trap2(f,1,2,2*n) - np.log(2)) for n in vec_n])

def erreur_simpson2(f,vec_n):
    return np.array([abs(simpson2(f,1,2,n) - np.log(2)) for n in vec_n])

# Fonctions de référence
def inv_2(x):
    return 1/x**2
def inv_3(x):
    return 1/x**3
def inv_4(x):
    return 1/x**4

# Tracé des erreurs
vec_n = np.arange(10,1001,10)
f = inv

plt.loglog(vec_n, erreur_rect2(f,vec_n),'x',label="erreur_rect2")
plt.loglog(vec_n, erreur_ptmilieu2(f,vec_n),'o',label="erreur_ptmilieu2")
plt.loglog(vec_n, erreur_trap2(f,vec_n),'+',label="erreur_trap2")
plt.loglog(vec_n, erreur_simpson2(f,vec_n),'.',label="erreur_simpson2")

vec_x = np.linspace(10,1001,100000)
plt.loglog(vec_x,inv(vec_x),label="1/x")
plt.loglog(vec_x,inv_2(vec_x),label="1/x^2")
plt.loglog(vec_x,inv_3(vec_x),label="1/x^3")
plt.loglog(vec_x,inv_4(vec_x),label="1/x^4")

plt.legend()
plt.show()
```